NLP for OpenFoodFacts

Jianrui SHI

https://colab.research.google.com/drive/1QMJCsTGkdbX_MdnYaKwuvVq7QPuPYOka?usp=sharing

Introduction

OpenFoodFacts can be considerated as a wikipedia for food! It contains more than 2.5 millions products but maybe all products are not perfectly described.
This time we analyzed and classified the ingredient list of these foods by NPL and identified the most similar foods. The project involves data cleaning, modeling, data visualization, food similarity calculation, and cluster analysis. A simple analysis has provided some insight into this dataset, but there are still many areas for improvement.

Data cleaning

The obtained dataset is a table with nutritional and compositional information about the food, and includes characteristics such as country. The dataset has 260,000 rows, and considering the analysis difficulty and RAM carrying capacity, I first selected 200,000 data for preliminary view (it is the maximum carrying capacity of the computer), The final 500,000 data was retained for the next step of analysis.

- Given that this dataset has more than 100 features, we first look at the number of null values.

```
#Check null data : there are lots of categories and many null dat
openfoods.isnull().sum().sort_values(ascending=True).head(50)
```

```
code                         0
url                          0
created_t                    0
created_datetime             0
last_modified_t              0
last_modified_datetime       0
states                       0
states_tags                  0
states_en                    0
completeness                 1
creator                      4
pnns_groups_2             3643
pnns_groups_1             3644
ecoscore_grade            4205
countries                 5432
countries_en              5435
countries_tags            5435
product_name             68455
last_image_t            371672
last_image_datetime     371672
energy_100g             412963
proteins_100g           421031
fat_100g                422774
```

- I found two categories about countries and looked at their values separately and found that they were not in a uniform format and included even more than 4000 countries. Given the limitations of NLP and my language, I decided to select the data from the United States (English) for analysis first.

```
# we have 2 country categories,the "contries_en" has 4000+ values
openfoods.countries_en.value_counts()
```

```
France                                                                      751205
United States                                                               531377
Germany                                                                     118383
Spain                                                                        94614
United Kingdom                                                               74534
                                                                               ...
Belgium,Francia                                                                  1
Francia,Suiza                                                                    1
French Guiana,Martinique                                                         1
fr:espagne-🇪🇸,fr:france🇫🇷,fr:portugal🇵🇹                                          1
Bosnia and Herzegovina,Bulgaria,Croatia,Montenegro,North Macedonia,Poland,Serbia 1
Name: countries_en, Length: 4286, dtype: int64
```

- Given the large dataset, can just delete the null and duplicate values.

```
# drop rows( null and duplicate Values)
new_openfoods = new_openfoods.dropna(axis=0, how='all')
new_openfoods = new_openfoods.drop_duplicates()
```

- Merge all U.S. data as the subsequent data set.

```
#there are too many null values and due to langue issues we choose data of US first
openfoods_us=openfoods[(openfoods['countries']=='United States')|(openfoods['countries']=='en:us')
```

- Now get a dataset with 50,000 rows, except for the product name and country, temporarily keep the other categories exist some null values.

```
new_openfoods.isnull().sum().sort_values()
```

```
product_name                0
countries                   0
energy_100g             38204
energy-kcal_100g        38251
carbohydrates_100g      39044
fat_100g                39202
proteins_100g           39315
sugars_100g             48936
salt_100g               67642
sodium_100g             67643
fiber_100g             145094
ingredients_text       203727
categories             225879
nutrient_levels_tags   235239
dtype: int64
```

```
len(new_openfoods)
```

```
524298
```

Ingredients Vectorization

I chose to further research the product by studying the food ingredients, performing tokenization, stemming, lemmazation, and removing stop words and various punctuation and numbers.

- Food ingredients is a data frame with index and values , for better research, changed it to list .

```
3] ingredients_us.head(10)
```

```
9      beta alanine, creatine hcl, ancient peat & app...
64     Bananas, vegetable oil (coconut oil, corn oil ...
65     Peanuts, wheat flour, sugar, rice flour, tapio...
126    Organic hazelnuts, organic cashews, organic wa...
127                                     Organic polenta
128    Rolled oats, grape concentrate, expeller press...
129                         Organic long grain white rice
130    Org oats, org hemp granola (org oats, evaporat...
131    Organic chocolate liquor, organic raw cane sug...
132    Organic expeller pressed, refined high oleic s...
Name: ingredients_text, dtype: object
```

- Cleaning all the test of this list .

```
def clean_text(text):
    if text is None:
        return ''
#remove punctuation and remove words containing numbers,take text lo
    text = str(text).replace("nan",'').lower()
    text = re.sub(r'\[.*?\]', '', text)
    text = re.sub(r'[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub(r'\w*\d\w*', '', text)
#tokenizer
    text_token = token.tokenize(text)
#lemaziner
    text_new = []
    for word in text_token :
        if (len(word) >= 1 and word not in STOPWORDS):
            word_lemma = lemma.lemmatize(word)
            word_stem = stemm.stem(word)
            text_new.append(word_stem)

    text_new =list(set(text_new))

    return text_new
```

- The 20 most frequent words in the food ingredients list are obtained by text cleaning and final word tokenization.

```
[ ] sorted(word_freq, key=word_freq.get, reverse=True)[:20]
```

```
['salt',
 'sugar',
 'flavor',
 'water',
 'acid',
 'oil',
 'natur',
 'corn',
 'milk',
 'flour',
 'sodium',
 'citric',
 'syrup',
 'color',
 'wheat',
 'starch',
 'contain',
 'less',
 'soy',
 'gum']
```

Modeling

For the resulting words, I modeled them using word2vec, which is a method of converting words into vectors. By calculating the distance of each word to calculate the word similarity, meanwhile, I got the map of food ingredients list.

- Build a dictionary by sorting the number of occurrences to get a dictionary of more than 5000 words.

```
[ ] len(w2v_model.wv.vocab)
```

```
5648
```

- Try to find the most similar word to a word（“oil”）

```
#find the most similar words in vocabulary of "oil"
w2v_model.wv.most_similar(positive=['oil'])
```

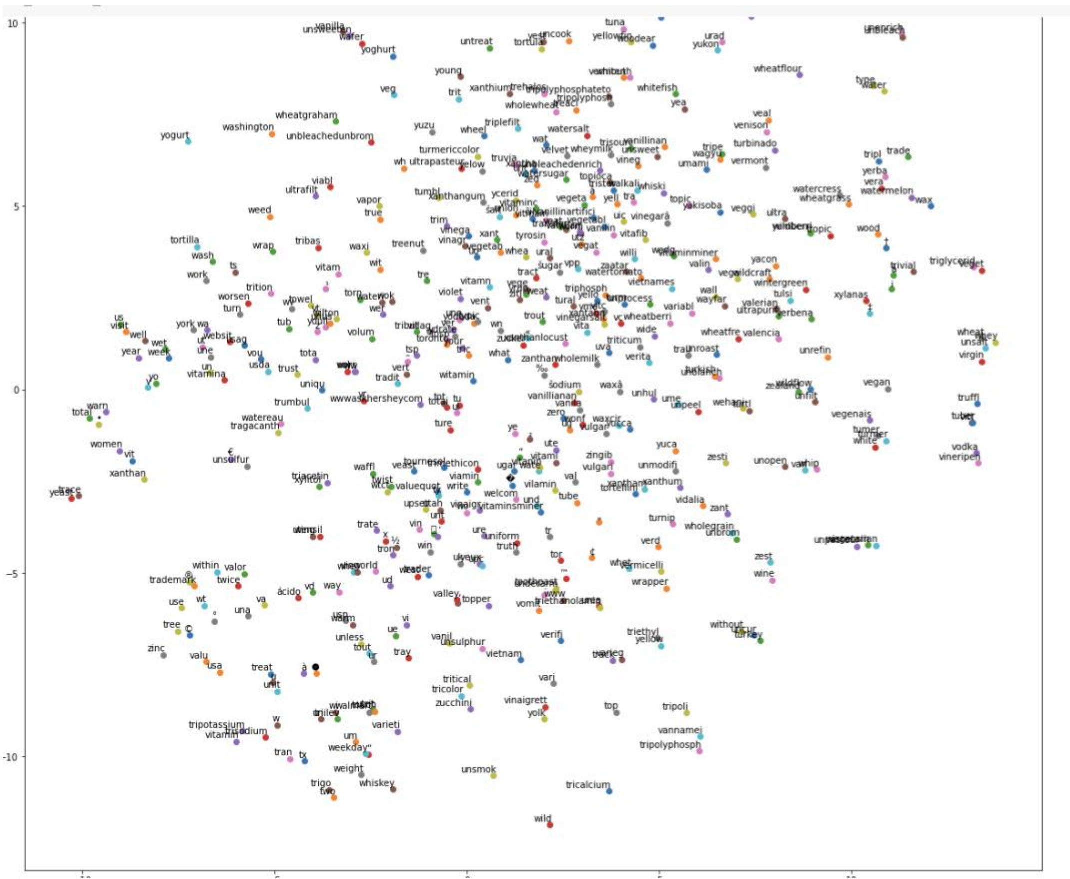
```
[('cornstarch', 0.3338833451271057),
 ('product', 0.3291834592819214),
 ('crouton', 0.3159681558609009),
 ('buttermilk', 0.3150302767753601),
 ('includ', 0.2947615385055542),
 ('potato', 0.2923141121864319),
 ('cauliflow', 0.28253111243247986),
 ('flake', 0.2824838161468506),
 ('herb', 0.27954649925231934),
 ('walnut', 0.27575528621673584)]
```

- Calculate the similarity of two values.

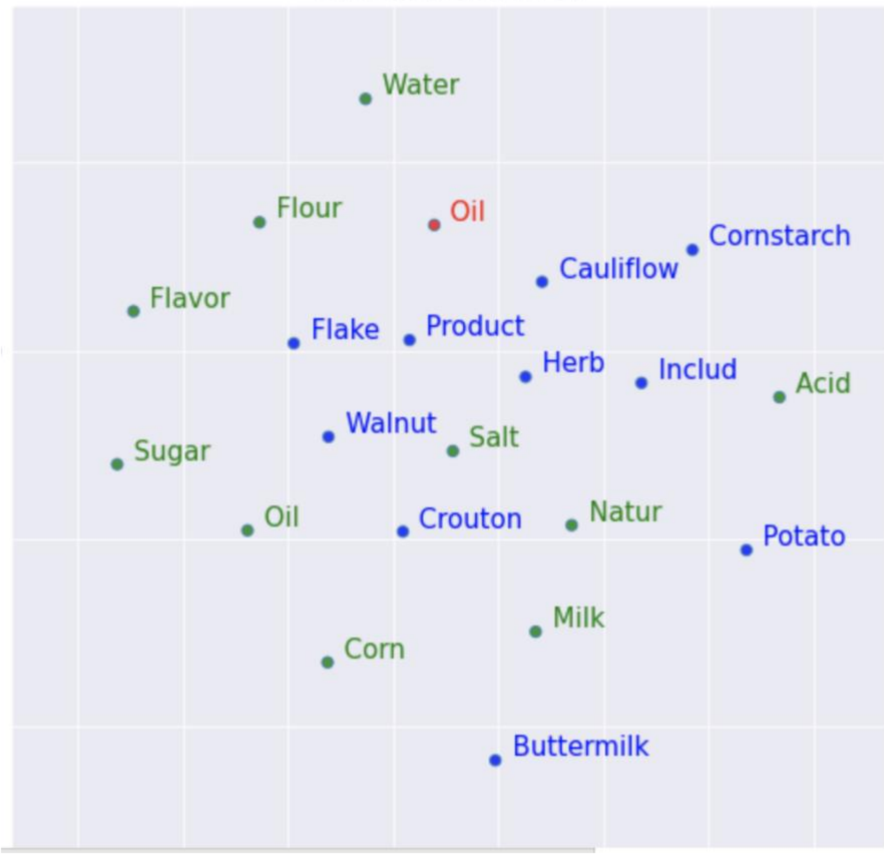
```
w2v_model.wv.similarity('chip','oil')
```

```
0.12528147
```

- Finally got the map of food ingredients list.



- The TSNE diagram visually displays similar words for the word "oil" and compares them with high-frequency words from the dictionary.



Similarly of Products

By calculating the similarity of the ingredient words in the food table, the similarity of two food products can then be obtained.

```
new_openfoods_us.head(10)
```

	product_name	ingredients_text
9	hyde icon	beta alanine, creatine hcl, ancient peat & app...
64	Banana Chips Sweetened (Whole)	Bananas, vegetable oil (coconut oil, corn oil ...
65	Peanuts	Peanuts, wheat flour, sugar, rice flour, tapio...
126	Organic Salted Nut Mix	Organic hazelnuts, organic cashews, organic wa...
127	Organic Polenta	Organic polenta
128	Breadshop Honey Gone Nuts Granola	Rolled oats, grape concentrate, expeller press...
129	Organic Long Grain White Rice	Organic long grain white rice
130	Organic Muesli	Org oats, org hemp granola (org oats, evaporat...
131	Organic Dark Chocolate Minis	Organic chocolate liquor, organic raw cane sug...
132	Organic Sunflower Oil	Organic expeller pressed, refined high oleic s...

- Enter the names of any two food products to get their similarity (based on food ingredients).

```
# find similarity of 2 products
from numpy import dot
from numpy.linalg import norm
def find_similarity(product1,product2):
    p1 = new_openfoods_us[new_openfoods_us.product_name == product1].index.tolist(
    p2 = new_openfoods_us[new_openfoods_us.product_name == product2].index.tolist(
    p1=p1[0]
    p2=p2[0]
    p_sen1 = clean_text(new_openfoods_us.at[p1,'ingredients_text'])
    p_sen2 = clean_text(new_openfoods_us.at[p2,'ingredients_text'])
    model = w2v_model.wv
    sen_vec1 = np.zeros(200)
    sen_vec2 = np.zeros(200)
    for val in p_sen1:
        sen_vec1 = np.add(sen_vec1, model[val])

    for val in p_sen2:
        sen_vec2 = np.add(sen_vec2, model[val])

    return dot(sen_vec1,sen_vec2)/(norm(sen_vec1)*norm(sen_vec2))
```

```
# Organic Salted Nut Mix and Organic Sunflower Oil
find_similarity('Peanuts','Peanuts')
```

```
1.0
```

- Enter the name of any 1 food product to get the product that is most similar to it.("Organic Salted Nut Mix")
  Some ingredients are not in the dictionary, so they cannot be calculated.

	product_name	similarity
3	Organic Salted Nut Mix	1.0
7	Organic Muesli	0.76971
5	Breadshop Honey Gone Nuts Granola	0.710573
9	Organic Sunflower Oil	0.694597
8	Organic Dark Chocolate Minis	0.565701
1	Banana Chips Sweetened (Whole)	0.340841
6	Organic Long Grain White Rice	0.313855
2	Peanuts	0.192369
0	hyde icon	NaN
4	Organic Polenta	NaN

K-means

Using K_means to further cluster analysis of food becoming, a map of food ingredients can be obtained.

- After embedding, perform clustering.

How to choose the number of clusters is a problem, I did not find a specific regulation, so I tried some random How to choose the number of clusters is a problem, I did not find a specific regulation, so I tried some random

```
#clustering
from sklearn.cluster import KMeans
clusters = 6
kmeans = KMeans(n_clusters=clusters, random_state=0).fit(embeddings)
```

- Map of Clustering

Conclusion

For this dataset, after selecting the sub-dataset, some features of the food ingredients could be found and the corresponding similarities were also calculated using different methods. But after modeling, I did not calculate the accuracy. Because I think there is still a lot of work to be done in the processing of the data, and the current data is not good enough to calculate the similarity.

Selection of data: This dataset has more than 100 categories, and I only selected food components for analysis, other categories will definitely have an impact on the results as well.

Data cleaning: I found that there are many special symbols (e.g. trademark R) in the maps generated by data visualization, in addition there are languages other than English (French), and there are also words that do not have any meaning (considered as spelling errors), all these uncleaned data affect the accuracy of the model.

Dictionary: I built a dictionary of more than 5000 words, but in the subsequent calculation of similarity, there are many words that are not in the dictionary, resulting in the inability to calculate, and will subsequently consider using incremental models

Running speed: The data set is too large, and the running time with RAM affects the training